Nominal Algebra

Aad Mathijssen Murdoch J. Gabbay

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

Process Algebra Meeting (PAM)
Centrum voor Wiskunde en Informatica (CWI)
Amsterdam
1st November 2006

Motivation

The λ-calculus

The λ-calculus:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t$$

Axioms:

$$\begin{array}{lll} (\alpha) & \lambda x.t = \lambda y.(t[x \mapsto y]) & \text{if } y \notin fv(t) \\ (\beta) & (\lambda x.t)u = t[x \mapsto u] & \\ (\eta) & \lambda x.(tx) = t & \text{if } x \notin fv(t) \end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Motivation

The λ-calculus

The λ-calculus:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t$$

Axiom schemata:

$$\begin{array}{lllll} (\alpha) & \lambda x.t & = \lambda y.(t[x \mapsto y]) & \text{if } y \notin fv(t) \\ (\beta) & (\lambda x.t)u & = t[x \mapsto u] & \\ (\eta) & \lambda x.(tx) & = t & \text{if } x \notin fv(t) \end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

t and u are meta-variables ranging over terms.

Motivation
The λ-calculus

The λ-calculus with meta-variables:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t \mid X$$

Axioms:

$$\begin{array}{llll} (\alpha) & \lambda x.X & = \lambda y.(X[x \mapsto y]) & \text{if } y \notin fv(X) \\ (\beta) & (\lambda x.X)Y & = X[x \mapsto Y] & \\ (\eta) & \lambda x.(Xx) & = X & \text{if } x \notin fv(X) \end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Motivation
The λ-calculus

The λ-calculus with meta-variables:

$$t \quad ::= \quad x \mid tt \mid \lambda x.t \mid X$$

Axioms:

$$\begin{array}{llll} (\alpha) & \lambda x.X & = \lambda y.(X[x \mapsto y]) & \text{if } y \notin fv(X) \\ (\beta) & (\lambda x.X)Y & = X[x \mapsto Y] & \\ (\eta) & \lambda x.(Xx) & = X & \text{if } x \notin fv(X) \end{array}$$

Free variables function fv:

$$fv(x) = \{x\} \qquad fv(tu) = fv(t) \cup fv(u) \qquad fv(\lambda x.t) = fv(t) \backslash \{x\}$$

Freshness occurs in the presence of meta-variables:
We only know if $x \notin fv(X)$ when X is instantiated.

Motivation
Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Motivation
Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Here:

- $\phi, \psi, P, Q, p, t, u$ are meta-variables ranging over terms.

Motivation
Other examples

In informal mathematical usage, we see equalities like:

- First-order logic: $(\forall x.\phi) \wedge \psi \quad = \forall x.(\phi \wedge \psi) \quad$ if $x \notin fv(\psi)$
- π-calculus: $(\nu x.P) \mid Q \quad = \nu x.(P \mid Q) \quad$ if $x \notin fv(Q)$
- μCRL/mCRL2: $\sum_x .p \quad = p \quad$ if $x \notin fv(p)$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $(\xi x.t)[y \mapsto u] = \xi x.(t[y \mapsto u]) \quad$ if $x \notin fv(u)$
- α-equivalence: $\xi x.t \quad = \xi y.(t[x \mapsto y]) \quad$ if $y \notin fv(t)$

Here:

- $\phi, \psi, P, Q, p, t, u$ are meta-variables ranging over terms.
- Freshness occurs in the presence of meta-variables.

Motivation

Formalisation

Question: Can we formalise binding and freshness

- in the presence of meta-variables
- in a direct way (without encodings)?

Motivation
Formalisation

Question: Can we formalise binding and freshness

- in the presence of meta-variables
- in a direct way (without encodings)?

Answer: Yes, using Nominal Terms (Urban, Gabbay, Pitts)

Motivation

Formalisation

Question: Can we formalise binding and freshness

- in the presence of meta-variables
- in a direct way (without encodings)?

Answer: Yes, using Nominal Terms (Urban, Gabbay, Pitts)

Question: Can we formalise equational reasoning about binding?

Motivation

Formalisation

Question: Can we formalise binding and freshness

- in the presence of meta-variables
- in a direct way (without encodings)?

Answer: Yes, using Nominal Terms (Urban, Gabbay, Pitts)

Question: Can we formalise equational reasoning about binding?

Answer: Yes, using Nominal Algebra...

Overview

Overview:

- Nominal terms
- Nominal algebra:
  - Definitions
  - Examples
- α-conversion and derivability
- Related work, with an application to choice quantification
- Results, conclusions and future work

Nominal Terms

Definition

Nominal terms are inductively defined by:

$$t \quad ::= \quad a \mid X \mid \mathsf{f}(t_1, \ldots, t_n) \mid [a]t$$

Here we fix:

- atoms $a, b, c, \ldots$ (for x, y)
- unknowns $X, Y, Z, \ldots$ (for t, u, ϕ, ψ, P, Q, p)
- term-formers $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$ (for λ, $\_\,\_$, $\forall$, $\wedge$, ν, $|$, $\sum$, $\_[\_ \mapsto \_]$)

We call $[a]t$ an abstraction (for the $x.\_$).

Nominal Terms
Sorts

We can impose a sorting system on nominal terms.

Sorts τ, inductively defined by:

$$\tau \quad ::= \quad \delta \mid [\mathbb{A}]\tau$$

Here:

- we fix base sorts $\delta, \delta', \delta'', \ldots$
- $\mathbb{A}$ is the set of all atoms $a, b, c, \ldots$
- $[\mathbb{A}]\tau$ represents an abstraction set:
  the set consisting of elements of τ with an atom abstracted

Nominal Terms
Sorting assertions

Assign to

- the set of atoms $\mathbb{A}$ a specific base sort δ
- each unknown X a sort τ, write X_τ
- each term-former f an arity $(\tau_1, \ldots, \tau_n)\tau$, write $\mathsf{f}_{(\tau_1,\ldots,\tau_n)\tau}$

Define sorting assertions on nominal terms, inductively by:

$$\frac{}{a : \delta} \qquad \frac{}{X_\tau : \tau} \qquad \frac{t : \tau}{[a]t : [\mathbb{A}]\tau}$$

$$\frac{t_1 : \tau_1 \quad \cdots \quad t_n : \tau_n}{\mathsf{f}_{(\tau_1,\ldots,\tau_n)\tau}(t_1, \ldots, t_n) : \tau}$$

Nominal Terms

Examples

Representation of mathematical syntax in nominal terms:

mathematics	nominal terms	
	unsugared	sugared
$\lambda x.t$	$\lambda([a]X)$	$\lambda[a]X$
$\lambda x.(tx)$	$\lambda([a]\mathsf{app}(X, a))$	$\lambda[a](Xa)$
$(\forall x.\phi) \wedge \psi$	$\wedge(\forall([a]X), Y)$	$(\forall[a]X) \wedge Y$
$(\nu x.P) \mid Q$	$\mid(\nu([a]X), Y)$	$(\nu[a]X) \mid Y$
$(\sum_x .p)$	$\sum([a]X)$	$\sum[a]X$
$t[x \mapsto u]$	$\mathsf{sub}([a]X, Y)$	$X[a \mapsto Y]$

Nominal Terms
Freshness

Definition:

- Call $a\#X$ a primitive freshness (for '$x \notin fv(t)$').
- A freshness context Δ is a *finite set* of primitive freshnesses.

Nominal Terms
Freshness

Definition:

- Call $a\#X$ a primitive freshness (for '$x \notin fv(t)$').
- A freshness context Δ is a *finite set* of primitive freshnesses.

Generalise freshness on unknowns X to terms t:

- Call $a\#t$ a freshness, where t is a nominal term.
- Write $\Delta \vdash a\#t$ when $a\#t$ is derivable from Δ using

$$\frac{}{a\#b}(\#\mathbf{ab}) \qquad \frac{}{a\#[a]t}(\#[]\mathbf{a}) \qquad \frac{a\#t}{a\#[b]t}(\#[]\mathbf{b}) \qquad \frac{a\#t_1 \quad \cdots \quad a\#t_n}{a\#\mathsf{f}(t_1,\ldots,t_n)}(\#\mathbf{f})$$

Examples:

$$\vdash a\#b \qquad \vdash a\#\lambda[a]X \qquad a\#X \vdash a\#\lambda[b]X$$

$$\nvdash a\#a \qquad \nvdash a\#\lambda[b]X \qquad a\#X \nvdash a\#Y$$

Nominal Algebra
Definition

Nominal algebra is a theory of equality between nominal terms:

- $t = u$ is an equality where t and u are of the same sort.
- $\Delta \rightarrow t = u$ is a judgement (for '$t = u$ if $x \notin fv(v)$').
  If $\Delta = \emptyset$, write $t = u$.

Nominal Algebra
Example judgements

Meta-level properties as judgements in nominal algebra:

- λ-calculus: $a\#X \rightarrow \lambda[a](Xa) = X$
- First-order logic: $a\#Y \rightarrow (\forall[a]X) \wedge Y = \forall[a](X \wedge Y)$
- π-calculus: $a\#Y \rightarrow (\nu[a]X) \mid Y = \nu[a](X \mid Y)$
- μCRL/mCRL2: $a\#X \rightarrow \sum[a]X = X$

And for any binder $\xi \in \{\lambda, \forall, \nu, \sum\}$:

- $a\#Y \rightarrow (\xi[a]X)[b \mapsto Y] = \xi[a](X[b \mapsto Y])$
- α-equivalence: $b\#X \rightarrow \xi[a]X = \xi[b](X[a \mapsto b])$

Nominal algebra

Theories

A theory in nominal algebra consists of:

- a set of base sorts
- a set of term-formers
- a set of axioms: judgements $\Delta \rightarrow t = u$

Nominal Algebra
LAM: the λ-calculus

A theory LAM for the λ-calculus with meta-variables:

- base sort $\mathbb{T}$
- term-formers λ, app and sub
  (recall that $t[a \mapsto u]$ is just sugar for sub$([a]t, u)$)
- axioms:

$$
\begin{array}{lllll}
(\alpha) & b\#X & \rightarrow & \lambda[a]X & = & \lambda[b](X[a \mapsto b]) \\
(\beta) & & & (\lambda[a]Y)X & = & Y[a \mapsto X] \\
(\eta) & a\#X & \rightarrow & \lambda[a](Xa) & = & X
\end{array}
$$

Nominal Algebra
LAM: instantiation of (β)

$$(\beta) \quad (\lambda[a]Y)X = Y[a \mapsto X]$$

Instantiation of (β):

Instantiation	Resulting judgement
	$(\lambda[a]Y)X = Y[a \mapsto X]$
$Y := b, X := c$	$(\lambda[a]b)c = b[a \mapsto c]$
$Y := a, X := c$	$(\lambda[a]a)c = a[a \mapsto c]$
$Y := a, X := c, a := b$	$(\lambda[b]a)c = a[b \mapsto c]$
$Y := (\lambda[b]Z)Y$	$(\lambda[a](\lambda[b]Z)Y)X = ((\lambda[b]Z)Y)[a \mapsto X]$

Nominal Algebra
LAM: instantiation of (η)

$$(\eta) \quad a\#X \to \lambda[a](Xa) = X$$

Instantiation of (η):

Instantiation	Resulting judgement
$X := a$	none: $\nvdash a\#a$
$X := b$	$\lambda[a](ba) = b$
$X := YZ$	$a\#Y, a\#Z \to \lambda[a]((YZ)a) = YZ$
$X := \lambda[a]Y$	$\lambda[a]((\lambda[a]Y)a) = \lambda[a]Y$
$X := \lambda[b]Y$	$a\#Y \to \lambda[a]((\lambda[b]Y)a) = \lambda[b]Y$

Nominal Algebra

FOL: first-order logic

A theory FOL for first-order logic with meta-variables, also called one-and-a-halfth-order logic:

- base sorts:
  - $\mathbb{F}$ for formulae
  - $\mathbb{T}$ for terms
- term-formers:
  - $\bot$, $\supset$, $\forall$, $\approx$ and sub for the basic operators ($\top$, $\neg$, $\wedge$, $\vee$, $\Leftrightarrow$, $\exists$ are sugar)
  - $p_1, \ldots, p_m$ and $f_1, \ldots, f_n$ for object-level predicates and terms
- axioms: ...

Nominal Algebra
Axioms of FOL

Axioms of one-and-a-halfth-order logic:

(MP) $\top \supset P \;=\; P$

(M) $((((P \supset Q) \supset (\neg R \supset \neg S)) \supset R) \supset T) \supset ((T \supset P) \supset (S \supset P)) \;=\; \top$

(Q1) $\forall[a]P \supset P[a \mapsto T] \;=\; \top$

(Q2) $\forall[a](P \wedge Q) \;=\; \forall[a]P \wedge \forall[a]Q$

(Q3) $a\#P \;\rightarrow\; \forall[a](P \supset Q) \;=\; P \supset \forall[a]Q$

(E1) $T \approx T \;=\; \top$

(E2) $U \approx T \wedge P[a \mapsto T] \supset P[a \mapsto U] \;=\; \top$

Nominal Algebra

SUB: a theory of capture-avoiding substitution

A theory SUB for capture-avoiding substitution with meta-variables:

$$
\begin{array}{rrcl}
(\mathbf{var}\mapsto) & a[a \mapsto T] & = & T \\
(\#\mapsto) & a\#X \to X[a \mapsto T] & = & X \\
(\mathbf{f}\mapsto) & \mathsf{f}(X_1, \ldots, X_n)[a \mapsto T] & = & \mathsf{f}(X_1[a \mapsto T], \ldots, X_n[a \mapsto T]) \\
(\mathbf{abs}\mapsto) & b\#T \to ([b]X)[a \mapsto T] & = & [b](X[a \mapsto T]) \\
(\mathbf{ren}\mapsto) & b\#X \to X[a \mapsto b] & = & (b\ a) \cdot X
\end{array}
$$

α-conversion
Problem

Formalising binding implies formalising α-conversion.

Idea: use theory SUB:

$$b\#X \rightarrow [a]X = [b](X[a \mapsto b])$$

α-conversion
Problem

Formalising binding implies formalising α-conversion.

Idea: use theory SUB:

$$b\#X \rightarrow [a]X = [b](X[a \mapsto b])$$

This destroys the proof theory:

- When proving properties by induction on the size of terms, you often want to freshen up a term using α-conversion.
- Freshening using the above α-conversion increases term size.

α-conversion
Solution

Solution: use permutations of atoms:

$$b\#X \rightarrow [a]X = [b]((a\ b) \cdot X)$$

α-conversion
Solution

Solution: use permutations of atoms:

$$b\#X \to [a]X = [b]((a\ b)\cdot X)$$

Redefine nominal terms:

$$t \quad ::= \quad a \mid \pi\cdot X \mid \mathsf{f}(t_1,\ldots,t_n) \mid [a]t$$

Here:

- we call $\pi\cdot X$ a moderated unknown
- write X when π is the trivial permutation **Id**
- instantiation of X to t in $\pi\cdot X$ gives us $\pi\cdot t$:

$$\pi\cdot a \equiv \pi(a) \qquad \pi\cdot(\pi'\cdot X) \equiv (\pi\circ\pi')\cdot X \qquad \pi\cdot[a]t \equiv [\pi(a)](\pi\cdot t)$$

$$\pi\cdot\mathsf{f}(t_1,\ldots,t_n) \equiv \mathsf{f}(\pi\cdot t_1,\ldots,\pi\cdot t_n)$$

Derivability of equalities

Write $\Delta \vdash_{\mathsf{T}} t = u$ when $t = u$ is derivable from the rules below, s.t.

- only assumptions from Δ are used
- each axiom used in $(\mathbf{ax}_{\Delta' \to t' = u'})$ is from theory T only

$$\frac{}{t = t}\,(\mathbf{refl}) \qquad \frac{t = u}{u = t}\,(\mathbf{symm}) \qquad \frac{t = u \quad u = v}{t = v}\,(\mathbf{tran})$$

$$\frac{t = u}{C[t] = C[u]}\,(\mathbf{cong}) \qquad \frac{a\#t \quad b\#t}{(a\ b)\cdot t = t}\,(\mathbf{perm})$$

$$\frac{\pi \cdot \Delta\sigma}{\pi \cdot t\sigma = \pi \cdot u\sigma}\,(\mathbf{ax}_{\Delta \to t = u}) \qquad \frac{\begin{array}{c}[a\#X_1, \ldots, a\#X_n] \qquad \Delta \\ \vdots \\ t = u\end{array}}{t = u}\,(\mathbf{fr}) \quad (a \notin t, u, \Delta)$$

Related work

Related work to Nominal Algebra (NA):

- Higher-Order Algebra (HOA)
- Cylindric Algebra and Lambda-Abstraction Algebra (CA/LAA)

These do not mirror informal mathematical usage like NA does:

- Binding and freshness are encoded:
  - by higher-order functions in HOA
  - by replacing t by $\mathsf{c}_i t$ to ensure $x_i \notin fv(t)$ in CA/LAA
- Reasoning about binding becomes different.
- Non-capturing substitution cannot be defined HOA/CA/LAA. It is the default notion of (meta-level) substitution in NA.

Choice quantification in μCRL/mCRL2

Axiom schemata

Axiom schemata for choice quantification (Groote, Ponse):

CQ1	$\sum_x p$	$= p$		if $x \notin fv(p)$
CQ2	$\sum_x p$	$= \sum_y p[x \mapsto y]$		if $y \notin fv(p)$
CQ3	$\sum_x p$	$= \sum_x p + p[x \mapsto d]$		
CQ4	$\sum_x (p + q)$	$= \sum_x p + \sum_x q$		
CQ5	$(\sum_x p) \cdot q$	$= \sum_x p \cdot q$		if $x \notin fv(q)$
CQ6	$\sum_x (d \rightarrow p)$	$= d \rightarrow \sum_x p$		if $x \notin fv(d)$

Note:

- infinite number of axioms
- no support for meta-variables

Choice quantification in μCRL/mCRL2

Axioms in Nominal Algebra

Axioms in Nominal Algebra for choice quantification:

$$
\begin{array}{lrcl}
\text{NCQ1} & a\#P \rightarrow \sum[a]P & = & P \\
\text{NCQ2} & a\#P \rightarrow \sum[a]P & = & \sum[b]P[a \mapsto b] \\
\text{NCQ3} & \sum[a]P & = & \sum[a]P + P[a \mapsto D] \\
\text{NCQ4} & \sum[a](P+Q) & = & \sum[a]P + \sum[a]Q \\
\text{NCQ5} & a\#Q \rightarrow (\sum[a]P) \cdot Q & = & \sum[a]P \cdot Q \\
\text{NCQ6} & a\#D \rightarrow \sum[a](D \rightarrow P) & = & D \rightarrow \sum[a]P
\end{array}
$$

Note:

- finite number of axioms
- direct correspondence with schemata
- NCQ2 is a lemma: α-conversion is built-in

Choice quantification in μCRL/mCRL2

Cylindric Algebra-style axioms

Cylindric Algebra-style axioms for choice quantification (Luttik):

CS1	$s_i s_j p$	$= s_j s_i p$	GC9	$s_i(d \rightarrow s_i p)$	$= c_i d \rightarrow s_i p$	
CS2	$s_i s_i p$	$= s_i p$	GC10	$s_i(c_i d \rightarrow p)$	$= c_i d \rightarrow s_i p$	
CS3	$p + s_i p$	$= s_i p$	GC11	$d_{ij} \rightarrow s_i(d_{ij} \rightarrow p)$	$= d_i p$	if $i \neq j$
CS4	$s_i(p + q)$	$= s_i p + s_i q$				
CS5	$s_i(p \cdot s_i q)$	$= s_i p \cdot s_i q$				
CS6	$s_i \delta$	$= \delta$				

Note:

- infinite number of axioms, one for each i and j
- related to schemata, but different: proofs become different
- existential quantification (c_i) is needed for the data language

Choice quantification in μCRL/mCRL2

Axioms in Higher-Order Algebra

Axioms in Higher-Order Algebra for choice quantification (Groote):

HCQ1	$\sum_x p$	$=$	p
HCQ2	$\sum_x F(x)$	$=$	$\sum_y F(y)$
HCQ3	$\sum_x F(x)$	$=$	$\sum_x F(x) + F(d)$
HCQ4	$\sum_x (F(x) + G(x))$	$=$	$\sum_x F(x) + \sum_x G(x)$
HCQ5	$(\sum_x F(x)) \cdot p$	$=$	$\sum_x F(x) \cdot p$
HCQ6	$\sum_x (d \to P)$	$=$	$d \to \sum_x P$

Note:

- finite number of axioms
- function variables F, G from data to process expressions
- variable condition on instantiation of HCQ1 and HCQ5

Nominal Algebra
Results

Results on nominal algebra:

- semantics in nominal sets
- proof system is sound and complete w.r.t. the semantics

Results on theory SUB (other work):

- omega-complete: sound and complete w.r.t. the term model
- equality $t = u$ is decidable

Results on theory FOL (other work):

- equivalent to first-order logic for terms without unknowns
- has an equivalent sequent calculus:
  - representing schemas of derivations in first-order logic
  - satisfies cut-elimination

Conclusions

Nominal algebra:

- is a theory of algebraic equality on nominal terms
- allows us to reason about systems with binding
- closely mirrors informal mathematical usage:
  - existing axioma schemata can be expressed directly
  - equational proofs carry over directly
  - natural notion of instantiation of meta-variables:

    informal notation: instantiating t to x in $\lambda x.t$ yields $\lambda x.x$
    nominal terms: instantiating X to a in $\lambda[a]X$ yields $\lambda[a]a$

Future work

Future work on nominal algebra:

- further develop theory on:
  - the λ-calculus
  - choice quantification in μCRL/mCRL2
  - π-calculus and its variants
  - reversibility
- formalise meta-level reasoning, meta-meta-level reasoning, ... a hierarchy of variables.
- develop a theorem prover

Further reading

- Murdoch J. Gabbay, Aad Mathijssen:
  Nominal Algebra.
  Submitted STACS'07.
- Murdoch J. Gabbay, Aad Mathijssen:
  Capture-Avoiding Substitution as a Nominal Algebra.
  ICTAC'06.
- Murdoch J. Gabbay, Aad Mathijssen:
  One-and-a-halfth-order Logic.
  PPDP'06.

Papers and slides of talks can be found on my web page:
http://www.win.tue.nl/∼amathijs